The role of markup in the digital humanities

Schmidt, Desmond

The Role of Markup in the Digital Humanities

Desmond Schmidt *

Abstract: *»Die Rolle der Textauszeichnung in den Digitalen Geisteswissenschaften«*. The digital humanities are growing rapidly in response to a rise in Internet use. What humanists mostly work on, and which forms much of the contents of our growing repositories, are digital surrogates of originally analog artefacts. But is the data model upon which many of those surrogates are based – embedded markup – adequate for the task? Or does it in fact inhibit reusability and flexibility? To enhance interoperability of resources and tools, some changes to the standard markup model are needed. Markup could be removed from the text and stored in standoff form. The versions of which many cultural heritage texts are composed could also be represented externally, and computed automatically. These changes would not disrupt existing data representations, which could be imported without significant data loss. They would also enhance automation and ease the increasing burden on the modern digital humanist.

Keywords: standoff markup, textual variation, cultural heritage texts.

1. Introduction

The Internet user population is now estimated, at the end of 2011, to be 32.7% of the world's population.[1] Whatever sub-percentage of that group are interested in their cultural heritage in the form of the writings of their literary greats and not-so-greats, historical documents such as journals and letters, fragments of papyrus, pottery, stone and metal with writing etc., that sub-percentage is growing and demanding online access to this information. There is also evidence that the number of jobs and grants in the digital humanities is increasing.[2] Projects of vast scope such as Dariah[3] and Europeana[4] are trying to capture this material and make it available via an ordinary web browser. All this places a burden of responsibility on digital humanists. Are the tools we currently have up to this increased strain? The Dariah and Bamboo[5] projects are most-

* Address all communications to: Desmond Schmidt, Information Security Institute, Queensland University of Technology, 126 Margaret Street, Brisbane, QLD, Australia; e-mail: desmond.schmidt@qut.edu.au.

1 "Internet World Stats," <http://www.internetworldstats.com/stats.htm>.

2 "Humanist Discussion Group, 25.321 growth of jobs in digital humanities," <http://www.digitalhumanities.org/humanist/>.

3 "Dariah: Digital Research Infrastructure for the Arts and Humanities," <http://www.dariah.eu/>.

4 "Europeana," <http://www.europeana.eu/portal/>.

5 "Project Bamboo," <http://www.projectbamboo.org/>.

ly about building repositories of data, and do not propose new technologies for content. What digital humanists primarily work on, however, are not the technologies to fetch objects but on the objects themselves.

1.1. Digital Surrogates

Digital surrogates are derived from original artefacts. An artefact might be a book, a manuscript, or an inscription, etc., usually in a library or a museum. A facsimile is one form of digital surrogate. Originals can be photographed and the images published online. This is useful for certain types of interaction, but especially when combined with digital transcriptions of the originals. Taken together, these two types of surrogate form the basis of everything that is built up: the interactive web sites, search indices, graphical user interfaces that allow the user to explore, annotate and edit the texts and their images. The question then becomes how good is the data model of the transcriptions upon which all these operations depend? Does it facilitate or inhibit the development of all that functionality? Can we share applications built upon it? Can we automate the process of creating a digital edition of a work or a set of documents? Can we make it easy to use so that training is minimised? All of these questions depend upon the data model we choose to represent the text, which until now has been embedded markup.

1.2. Plan of the Paper

The rest of this paper is structured as follows. Section 2 is a brief history of markup and its role in the humanities. Section 3 identifies six key weaknesses in the existing markup data model. Section 4 then describes an alternative model that integrates markup in standoff form with the versions of which cultural heritage texts are often composed. Section 5 draws some conclusions and describes future work and current collaborations.

2. History of Markup

Markup evolved from the practice of adding mark-up instructions to documents being prepared for printing.[6] In early word processors like runoff,[7] created in the 1960s, dot-commands were inserted into the text on separate "control-lines" such as ".indent 5". In 1973 the GML language respecified these specific instructions as generalised names like ":h0.Chapter 1", which were then proc-

[6] *The Oxford English Dictionary Online* (Oxford: OUP, 2000), s.v. 'mark up'.

[7] J. Salzer, "TYPSET and RUNOFF, Memorandum editor and type-out commands 1964," <http://mit.edu/Saltzer/www/publications/CC-244.html>.

essed by an external "profile" of computer code.[8] SGML, standardised by IBM in 1986,[9] was an evolution of GML that allowed the document structure to be defined via a text file called a DTD.[10] SGML was later revised by the W3C, with support from Microsoft, as XML in 1998.[11] The attraction of XML for Microsoft was its potential in implementing "web services". These were programs that could communicate over the Web by sending XML messages, in effect verbs, describing what a service does, as part of Microsoft's .Net initiative.[12] This style of web service has since become less popular, and is being gradually replaced by REST (representational state transfer), a simpler noun-based methodology that describes what resources a service has, similar to old-style static web pages.[13] This drive for simplification recently led James Clark, the technical lead developer of XML, to disown XML in favour of JSON (Javascript object notation).[14] The picture that this short history paints is thus one of continuous change and evolution of the markup model.

2.1. Markup in the Digital Humanities

Humanists have always followed these developments closely. The Brown corpus in the 1960s used embedded formatting codes based on those used by the US Patent Office.[15] Other early examples are the use of "discriminants" to structure texts in Silva and Bellamy's suite of programs to process "language data",[16] and the embedded COCOA tags as used in the Oxford Concordance Program[17]. The Text Encoding Initiative set out to establish an interchange format to resolve what seemed at the time a chaotic explosion of proprietary formats on CD-ROM and other digital media that threatened to make sharing of

[8] "GML Starter Set User's Guide," <http://publibfp.boulder.ibm.com/cgi-bin/bookmgr/BOOKS/dsm04m00/CCONTENTS>.

[9] *ISO 8879. Information processing – Text and office systems – Standard Generalized Markup Language (SGML)* (Geneva: ISO, 1986).

[10] C. Goldfarb, "The Roots of SGML – A Personal Recollection," <http://www.sgmlsource.com/history/roots.htm>.

[11] "XML," <http://en.wikipedia.org/wiki/XML>.

[12] J. Clark, November 24, 2010, "James Clark's Random Thoughts," <http://blog.jclark.com/2010/11/xml-vs-web_24.html>; T. Anderson. "Introducing XML," <http://www.itwriting.com/xmlintro.php>.

[13] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures" (PhD dissertation Irvine, 2000).

[14] J. Clark, *ibid.*

[15] W. N. Francis and H. Kucera. "BROWN CORPUS MANUAL. Brown: Providence, 1964," <http://khnt.hit.uib.no/icame/manuals/brown/index.htm>.

[16] G. Silva and C. Bellamy, *Some Procedures and Programs for Processing Language Data* (Melbourne: Monash University, 1968).

[17] S. M. Hockey and I. Marriott, *Oxford Concordance Program Version 1.0 User's Manual* (Oxford: Oxford University Computing Service, 1980).

texts in the humanities impossible.[18] The result of the conference at Poughkeepsie, New York in 1987 was the drawing up of a set of Guidelines for Electronic Text Encoding and Interchange, first published as P3 in 1990.[19] It was initially based on SGML, later XML (P4, 2002).[20] It is often described as "the de facto standard for the encoding of electronic texts in the humanities academic community".[21] Although not exactly mandated by the American NEH (National Endowment for the Humanities), who originally funded it, applicants for scholarly editions grants are encouraged to use TEI, and if they do not, they must justify their decision.[22] The list of projects using TEI is admittedly impressive;[23] it is obvious that this technology: embedded XML markup, is the norm for encoding texts in the digital humanities.

3. The TEI Guidelines and the Standard Data Model

There are now, with the addition of the genetic edition tags, 544 elements defined in the TEI Guidelines.[24] Judging by the list of projects using the TEI and the institutions supporting it, one could say that it is of interest mostly to digital humanists, and to a lesser extent, linguists. Nevertheless, many of the tags represent programming information, social science codes and metadata.

The following sections describe six key problems with the TEI and the embedded markup model on which it is based.

3.1. Size

With its 544 tags the TEI Guidelines have already become very difficult for new users to work out if there is a tag for a particular feature. Michael Sperberg-McQueen warned early on in the TEI's history that the number of potential tags for labelling texts was "unbounded".[25] But what are the other consequences of size? One is the increasing difficulty of writing any software to

[18] N. Ide and C. M. Sperberg-McQueen, "Proposal for Funding for An Initiative to Formulate Guidelines for the Encoding and Interchange of Machine-Readable Texts," <http://projects.oucs.ox.ac.uk/teiweb/Vault/SC/scg02.html>.

[19] *Text Encoding Initiative, Guidelines for Electronic Text Encoding and Interchange,* ed. C. M. Sperberg-McQueen and L. Burnard (Chicago, Oxford: 1990).

[20] TEI, "P4 Guidelines," http://www.tei-c.org/Guidelines/P4/.

[21] "Wikipedia, Text Encoding Initiative," <http://en.wikipedia.org/wiki/Text_Encoding_Initiative>.

[22] NEH, "Scholarly Editions and Translations," <http://www.neh.gov/files/grants/scholarly-editions-dec-8-2011.pdf>, 8.

[23] TEI, "Projects Using TEI," <http://www.tei-c.org/Activities/Projects/>.

[24] TEI, "P5: Guidelines for Electronic Text Encoding and Interchange. Elements," <http://www.tei-c.org/release/doc/tei-p5-doc/en/html/REF-ELEMENTS.html>.

[25] C. M. Sperberg-McQueen, "Text in the Electronic Age Textual Study and Text Encoding, with Examples from Medieval Texts," *Literary and Linguistic Computing,* 6.1 (1991): 36.

encompass all of it. Although there are now many specialised programs that use the TEI Guidelines as an input or output format[26] they mostly work on small and different subsets of the tags, and hence are not interoperable.

3.2. Usability

One consequence of embedding markup into the text is the necessity to see all the tags and their attributes when editing it. The problem is that many codes, e.g. for variants, for metadata tags and joins, etc. have no clear mapping between elements and formatting. In a GUI (graphical user interface) you can make text starting with "<hi rend="italic">" appear in italics. But what do you do with "<link target="#n3.284 #r3.284 #L3.284"/>"? Size also comes into play here. Building an easy-to-use editor for TEI means finding some way to represent *all* the tags in the scheme in a graphical way. The oXygen editor, a proprietary product, does as good a job as any,[27] but it cannot render the more complex encoding as anything except raw XML. Admittedly, complex markup may be rendered as simplistic HTML, and displayed on screen after a program has digested and transformed it. But the problem of how to develop a simplistic interface for the editor of the original XML remains, and appears to be unsolvable.

3.3. The Overlap Problem

Most digital humanists have heard of the "overlapping hierarchies" problem made famous by the 1993 paper by Renear, Mylonas and Durand.[28] What was noticed very early on in the application of generalised markup languages to humanities texts was their apparent inability to represent anything except tree-structures;[29] whereas what humanists often wanted to record seemed to be non-tree structures. Start and end tags needed to overlap in ways that led to ugly hacks in SGML and later XML that no one liked. But what is the extent of the problem? Although overlap is common in digital humanities texts it occurs in three quite distinct scenarios:

1) Variation of content and structure
2) Overlay of multiple perspectives or markup sets, e.g. between a reference system and a formatting structure

---

[26] TEI, "Category: Tools," <http://wiki.tei-c.org/index.php/Category:Tools>.

[27] "Oxygen XML Editor," <http://www.oxygenxml.com/>.

[28] A. Renear, E. Mylonas, D. Durand, "Refining our Notion of What Text Really Is: The Problem of Overlapping Hierarchies," <http://www.stg.brown.edu/resources/stg/monographs/ohco.html>.

[29] D. Barnard, R. Hayter, M. Karababa, G. Logan and J. McFadden, "SGML-Based Markup for Literary Texts: Two Problems and Some Solutions," *Computers and Humanities* 22 (1988): 265-76.

3) Overlap of individual start and end tags within a single markup perspective.

It is unclear that there are many cases of type 3. The case mentioned by Barnard of the line in a Shakespeare play split between two speakers[30] is a classic problem that might be redefined as one of schema definition rather than overlap (i.e. by allowing speech inside line as well as line inside speech). DeRose's nested quotation in the Old Testament that cuts across the verse structure[31] might likewise be recast as a conflict between formatting of quotations and the biblical verse structure, i.e. class 2 overlap. One legitimate example, however, is Bauman's quotation that overlaps line-breaks.[32] But on the whole these cases do not seem to be very numerous.

Class 1 overlap can be dealt with successfully via an external method for handling versions, i.e. outside of markup. This will be explained in the next section. This greatly reduces the overlap problem, but does not eliminate it. Even if class 3 can be ignored (and it probably cannot), we are still left with the problem of how to deal with class 2 overlap.

3.4. Interlinking

Embedded markup languages always create a primary tree structure.[33] Getting around this limitation with embedded XML usually relies on the creation of links between elements via attributes. One might use this technique, for example, to link overlapping variants through "joins",[34] or to interconnect an arbitrary set of elements for any other purpose. This extension to XML on the surface seems attractive, and in the TEI Guidelines linking is used extensively to get around the limitation of tree-structures. As well as explicit linking attributes for certain elements, *any* element may be joined to any other through the use of global attributes.[35] But there are serious drawbacks with this general approach:

1) The grammar of the markup language does not normally support verification that the link target or targets exist.[36]
2) Several interlinked elements may accidentally form a directed cycle, so that a program following them may never terminate.

---

[30] Barnard et al., *ibid.*

[31] S. DeRose, "Markup Overlap: A Review and a Horse" (paper presented at Extreme Markup Languages, Proceedings of Extreme Markup Languages, 2-6 August 2004, Montreal, Canada) <http://conferences.idealliance.org/extreme/html/2004/DeRose01/EML2004DeRose01.html>.

[32] S. Bauman, "TEI Horsing Around" (paper presented at Extreme Markup Languages Conference, 2005), <http://conferences.idealliance.org/extreme/html/2005/Bauman01/EML2005Bauman01.html>.

[33] D. Schmidt, "The Inadequacy of Embedded Markup for Cultural Heritage Texts," *Literary and Linguistic Computing* 25.3 (2010): 344.

[34] TEI, P5, 16.7.

[35] TEI, P5, 16.

[36] S. Bauman, *op. cit.*

3) Information is stored on two levels: the structure of the markup and the structure of the links, which must be processed separately.
4) Easy to use interfaces for editing interlinked elements do not exist; the user is left to struggle with the increased complexity.
5) There are no constraints on the links that can be created. One could easily create a structure that would not be processable by computer.

3.4.1. Travelling Salesman

Point 5 needs some expansion to point out the seriousness of the problem. A familiar example to software engineers is the travelling salesman who is asked to sell his wares in 25 cities.[37] To save money, he asks a travel agent to find the lowest cost flights between each pair of cities, then to work out the cheapest itinerary that visits each city once. After working out the flight costs, the travel agent asks a programmer to find the cheapest itinerary. His program considers 10 million itineraries per second. It runs and runs but does not terminate. Why? Because there are approximately 1.5×10^{25} possible itineraries, and considering even 10 million per second will take a staggering *49 billion years.*

The analogy with XML is that arbitrary links between elements are like the flights between cities. The itineraries represent the visits our program makes between the elements, following those links. It may be that our network of interconnected elements has been written by a program, and we are in a position to prove that it is processable, but if we allow humans to edit the XML and to create arbitrary networks of links it is not in general possible to verify that the result will be computable.

So texts encoded in the TEI Guidelines, if they use the linking facilities, may not be processable in the way intended, because in general linked elements are not verifiable and pose a hard computing problem. In the absence of documentation of how the links are to be used in each particular case, and without knowing their correctness and tractability, the task of writing an import or translation program for such documents appears to be at least very challenging.

3.5. Variation

It was noted above that type 1 overlap was caused by textual variation. By variation is meant any differences in the text or structure that arise from changes made to an original document, or differences between versions of one work, e.g. editions of a book. When creating a digital surrogate of a work it is an advantage to put all the variation into one text, where variants can be easily compared and the text in common can be specified only once.[38] But when using

[37] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (New York: W. H. Freeman, 1979).
[38] H. W. Gabler, "The Primacy of the Document in Editing," *Ecdotica* 4 (2007): 207.

embedded markup surrogates this is rarely possible. The problem is that even more than a few variants makes TEI markup constructs like "parallel segmentation"[39] overly complex, inaccurate and repetitious. The only way around this limitation is to encode each physical document as a separate file. But then the XML versions need to be compared or collated, which leads to the following problems:

a. The computed differences are fragments of the original and may not be well-formed XML,[40] so one has to strip out the tags either after the comparison is made or before, which throws away information. It will not be possible, therefore, to compare any formatting property such as italics that is present in one version and absent or changed in another, or even to display such properties at all reliably.
b. Comparison between variants embedded in separate documents is very difficult.
c. Using markup to record variation is a manual process, and has to deal with the most complex markup in the TEI Guidelines. It is thus very time-consuming and fails to harness the power of the computer to perform a task that is entirely automatable.

3.6. Interoperability

The desire to standardise markup, as in the TEI, has always been – at least in part – connected with the desire to share software. And sharing software is what we all want because development and maintenance costs are so high. Free software tools can be simply downloaded by programmers and reused to produce new products. Although digital humanists can also reuse other people's software, they appear less able to reuse software written by other digital humanists. There seem to be two main reasons for this:

3.6.1. Subjectivity of Markup

That the process of marking up a text in the humanities is subjective is not in dispute.[41] Subjectivity is present in the selection of codes the humanist makes to record the features he/she wants. The existence of a feature like underlining may not be in dispute, but it may not be of interest to the encoder. Likewise each transcriber understands the text and the codes that apply to it differently.

---

[39] TEI, *P5,* 12.2.3.

[40] "Extensible Markup Language (XML) 1.0 (Fifth Edition)," <http://www.w3.org/TR/2008/REC-xml-20081126/#sec-well-formed>.

[41] E.g. P. Eggert, "Text-encoding, Theories of the Text, and the 'Work-Site,'" *Literary and Linguistic Computing* 20.4 (2005): 425-35.

The result of such a process can only be subjective, and like a fingerprint, encodings of the same analog text by two people are almost never the same.[42]

3.6.2. End-Result Related Information

Generalised markup systems like HTML are supposed to separate structure from presentation. In reality this is only partly possible. A close examination of a typical HTML file will show that formatting instructions in an external stylesheet are tightly bound to specific elements. They are put in another file so they can be edited conveniently, but such a pretend separation of form and presentation is just a programmatic trick. Likewise, in many humanities texts it is almost impossible to avoid embedding information about the end-result when using "generalised" markup codes: programmatic instructions about how to link one text to another, information to assist collation or to refer to areas on an image, and in the latest TEI Guidelines there are new elements devoted to the precise page layout of the text of a genetic edition using grid coordinates and dimensions.[43] This kind of embedded markup uses the mechanisms of generalised markup to record specific features of the end-result. One can, however, argue that all embedded markup in the humanities has a similar purpose. What you want to get out of a digital surrogate is connected with what you choose to put into it.

3.6.3. Standardisation and Interoperability

Although digital humanists may be content to transcribe what they see in the original or facsimile without any clear idea about what their codes may eventually be used for, in practice all codes are either connected with the software that processes them, or they are unused. When software is written that depends on these subjective and end-result related markup codes it cannot be of use to someone who intends to use that software on another text. Some examples should make this point clearer.

Example 1: Two groups are working on Shakespeare. The first is studying his language and use linguistic tools to analyse it. They mark up the text automatically using POS (part-of-speech) taggers. They ask another project for more Shakespeare texts. This other project is producing an online edition of Shakespeare, marking up the text manually with act, scene, speech, stage directions, lines and paragraph structures. They also would like to interchange texts with the linguists. But none of the embedded markup of either group is usable by the other, and so all must be stripped out before the texts can be "interchanged".

---

[42] S. Bauman, "Interoperability vs. Interchange" (paper presented at Balisage: The Markup Conference, Montreal, Canada, August 2-5, 2011), doi:10.4242/BalisageVol7.Bauman01.

[43] TEI, P5, 11.

Example 2: Two German literature projects are working on an edition of a philosopher and a poet. Apart from the occasional "paragraph" tag, most of the embedded tags are not understood by the software of either project. Since stripping out the incompatible codes would create non-well-formed documents, and since the codes do not map to equivalent structures, all tags must be stripped out before interchange can take place. For example, the philosophical structure "remark" has no analogue in poetry. Likewise "stanza" is a concept alien to philosophical texts.

Interoperable software thus cannot be built on the basis of embedded tags that are subjectively defined, subjectively chosen and subjectively applied. Truly interoperable formats like SVG (scalable vector graphics) are real standards based on a one-for-one mapping between language constructs and program functions.[44] An instruction to draw a polygon does exactly that in each compliant SVG application. Humanistic data is simply not like that. It is interpretative, not functional.

3.6.4. Removing Markup

It may seem that in order to reuse a text with embedded markup one only has to strip out the markup. But first reflect that removing the markup means removing the very thing one hoped to standardise. Some software operated upon those codes; removing them is a declaration of intent to re-encode and to write new software.

Perhaps the most difficult problem about removing markup from text is the treatment of white space. In XML white space may be part of the content or the markup. White space is used to neatly lay out XML so we can read it on screen: the tabs, spaces and new lines "pretty print" the hard-to-read code. Humans, however, find it difficult to distinguish between these two very different types of white space. After all, they look exactly the same. Is a space between two marked up syllables, sentences, lines etc. significant or just XML layout? A program that strips markup codes has to know the exact syntax of that particular file in order to make the correct decisions in such cases, but how can the human editor who created it be expected to keep such complex information in mind while editing it?

Another problem is the practice of encoding alternatives like <sic> and <corr>, <abbr> and <expan>, <add> and <del>, <app> and <rdg> etc., and nested combinations thereof. Stripping out these naively will make nonsense of content that made sense in XML, and stripping them out intelligently will split one file into several, or favour one version over many.

---

[44] W3C, "Scalable Vector Graphics (SVG) 1.1 (Second Edition)," <http://www.w3.org/TR/SVG/>.

Another problem is the inflexibility of stripping software. Humans are not robots and inconsistencies in markup practice will translate into errors during the extraction of plain text. For these reasons stripping out markup from a text is a lot harder than it seems at first, and is never guaranteed to be perfect.

4. How to Design a Solution

Imagine for a moment that all the problems enumerated above have been solved at some point in the future. What constraints does the mere existence of such a solution place on its design that already tells us how it will look? Figure 1 is an attempt to resolve all the problems without looking at how they can be implemented. This is important, because what software and human interface designers tell us about writing an application is to first determine what the users want. At the design phase "engineers are not allowed to say it cannot be implemented (even when they know it cannot)".[45] This is because what seems impossible (but is not really) might get in the way of building what is needed. Linus Torvalds says something similar: "The code itself is unimportant; the project is only as useful as people actually find it."[46] One way of applying this principle to digital humanities projects is not to make adherence to standards a prerequisite that might shut out possible ways to satisfy users' needs.

But what are the users' needs? A good starting point would seem to be the solution of the problems described above. If a solution can be built on that it would provide a good example implementation to elicit further user responses. Without this first step, it is unlikely users will be able to articulate accurately enough the design of a fully finished application.

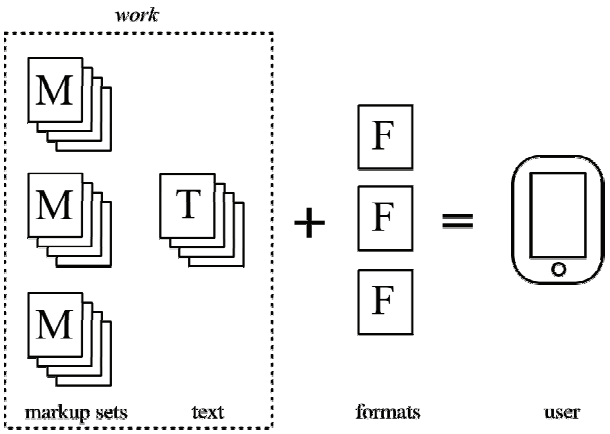
What can be seen in Figure 1 is that the markup of a work is outside the text and divided into sets, and both are subdivided into versions. On the right, formats give the markup and text a concrete visualisation that users can interact with via the browser. And browser interaction is all that is required, since as argued above, use of the Internet is becoming pervasive worldwide. Since every electronic edition will need its own look and feel it must be customisable at every level. Dividing markup into sets greatly increases its flexibility, since each set represents some view or layer of markup such as a reference system, a set of links to external images or another text, a set of annotations or structural markup. Each combination of markup sets on the left will structure the text differently and may result in portions of it being temporarily removed. Selecting a format from the set on the right then customises that structured data further for viewing or interaction by the user.

---

[45] B. Tognazini, *TOG on Interface* (Reading: Addison-Wesley, 1992): 27.

[46] S. Vaughan-Nichols, September 26, 2011, "Linus Torvalds's Lessons on Software Development Management," <http://h30565.www3.hp.com/t5/Feature-Articles/Linus-Torvalds-s-Lessons-on-Software-Development-Management/ba-p/440>.

Figure 1: Abstract Model for Digital Humanities Web-based Texts



4.1. Automating Variation

One of the key problems with embedded markup is its weakness in being able to represent textual variation. But biologists have long used automated techniques for computing differences between texts that exhibit the same insertions, deletions, variants and transpositions present in humanities texts.[47] Automating computation of variation would solve two of the key problems described above:

a. the problem of overlap arising from textual variation, and
b. the slowness of manual markup

Only through automation can the increasing demands of an ever-growing web community to have access to our textual cultural heritage be met. And that heritage is largely in the form of works existing in multiple versions: edited drafts of short stories and novels, successive print editions, multiple manuscripts of ancient and medieval works with significant differences. Even a completely clean one-version printed text is likely to contain mistakes and so may need correcting via a second version.

The nmerge program, previously described,[48] already performs that function. It primarily merges texts rather than collates. Each different expression of a work or part of a work is combined into a single digital entity that can be searched, and its versions can be rapidly compared or extracted for archiving or

[47] J. Bourdaillet and J.-G. Ganascia, "MEDITE: A Unilingual Textual Aligner" (paper presented at the 5th International Conference on Natural Language Processing, FinTAL 2006, Turku, Suomi) *Lecture Notes in Artificial Intelligence* 4139 (2006): 458-69.

[48] D. Schmidt, (paper presented at Balisage: The Markup Conference July 20-26, 2009) *Balisage Series on Markup Technologies* 3 (2009). doi:10.4242/BalisageVol3.Schmidt01.

reading. The key advantage of this approach over collation-based approaches like CollateX[49] is that it uses the faster byte-by-byte comparison technique of suffix trees.[50] This advance was first applied to humanities texts in the MEDITE viewer, though only for two versions at a time.[51] Nmerge then extended this capability to multiple versions. Collation programs, on the other hand, are slower, only compare word by word, and are more likely to be led astray by large transpositions or insertions. Merging via suffix trees, on the other hand, can correctly and rapidly identify heavily revised versions such as Charles Harpur's *Creek of the Four Graves,* which has less than 20% similarity between two of its versions. Manual markup of variations between two such radically different texts would be unthinkable, and the apparatus would be so voluminous as to be unusable. But by simply comparing merged copies, as can be seen on the Digital Variants website side-by-side,[52] this complex information is made entirely comprehensible to the user.

What this means is that using markup to record variation is now unnecessary. And with that a great deal of complexity in the markup of cultural heritage texts simply disappears. Since even a large number of versions can be successfully merged, manual editing of texts to determine differences that might have taken years can now be performed in seconds, once transcriptions of each version have been obtained. What results from the merging process is an entity that corresponds to a "work", and all its expressions, which is what we intuitively want to store in our digital repository, not the complexity of individual documents that the user does not understand, or needs to follow numerous links to find.

4.1.1. Objections

Two objections have been raised against this approach to automatically compute versions versus manual encoding:

1) Manual encoding allows the editor to control what is a variant of what, whereas automatic computation might get it wrong.
2) Specifying second level and third level corrections in a manuscript as all belonging to separate layers distorts the text. Displaying corrections "diplomatically" i.e. inline with crossing-out is better way to represent the correct succession of changes.

---

[49] "CollateX," <http://collatex.sourceforge.net/>.
[50] E. Ukkonnen, "Online Construction of Suffix Trees," *Algorithmica* 14 (1995): 249-60.
[51] J. Bourdaillet and J.-G. Ganascia, *op. cit.*
[52] D. Fiomonte, "Digital Variants," <http://www.digitalvariants.org/texts>.

4.1.2. Answers

1) It is true that automatic merging may possibly misalign and thus misrepresent a correction. A hypothetical example would be the replacement of an entire word with a small variation of that word. Automatic alignment would then flag only the one-letter difference as changed. Such problems, however, are rare. Consider instead how many times manual markup is forced to hack the encoding to represent things it cannot, such as overlapping variants, or transpositions. Automatic alignment will never be perfect, but it is still many times more accurate than manual markup. If the user is really interested in such small differences he/she can always consult the facsimile.
2) This objection confuses presentation with the data model. Automatic alignment can represent anything that embedded markup can for versions, and far more. When you specify a sequence of alterations such as "<subst> ~~dog~~<add>cat</add></subst>" using TEI, you are effectively assigning "dog" the status as the first version and "cat" as the second. Merging does exactly the same thing. If "cat" actually belongs to a third layer (instead of the second) then the editor need only create a third version containing "cat" and merge it with the other versions. If the user needs a diplomatic rendition of the text, this can be generated from the multi-version document as a separate view.

4.2. Markup Outside the Text

The second half of this redesign is the removal of the remaining embedded markup (unrelated to versions) to a standoff form. There have been three proposals for achieving this.

4.2.1. Standoff Markup

Standoff markup was apparently invented by the linguist Ralph Grisham, in the Tipster II document format.[53] In standoff markup tags are removed from the text and stored in a separate file, together with their offsets into the plain base text. In this way any markup-set can be later combined with the plain text to produce a normal SGML or XML document. This has the advantage of allowing alternative competing hierarchies to be chosen for different tasks, e.g. for different types of linguistic analysis. The text is also simplified by having the markup removed. However, this doesn't change the status of the markup as a hierarchy of tags. Also markup sets cannot be combined because the simultaneous assertion of two competing grammars and their overlapping tags would render the result not well-formed.

[53] Grisham, R. "Tipster Phase II Architecture Design Document (Strawman Architecture) Version 1.10" <ftp://cs.nyu.edu/pub/nlp/tipster/100.tex>.

4.2.2. Semantic Markup

The second proposal is to use RDF (resource description framework) or its more extended form, OWL (web ontology language), as a replacement for standoff markup.[54] RDF has the advantage of already being a widely used standard. But it was designed to inter-relate resources on the web via logical "triples" consisting of subject-predicate-object as in "My dog"-"has the name"-"Rover". EARMARK thus seems a heavyweight solution for the simple task of describing textual properties. In principle it is no different from standoff markup, except that it is expressed using OWL notation, so that, for example, overlaps can be discovered in SWRL (semantic web rule language).[55] However, it does not seem to be intended as a storage format.[56]

4.2.3. Standoff Properties

Classic standoff markup is too limiting. Semantic markup was designed for a different task. Standoff properties, on the other hand, are as lightweight as standoff markup and can represent overlap without difficulty. Standoff properties are not merely tags stripped from texts that will later be put back in the same place. Instead, each standoff property asserts a name over a range of text. It doesn't belong to a grammar and so can freely overlap with other properties. Sets of standoff properties can be freely intermingled because each property is autonomous. However, this is hardly an original idea. Thaller's extended strings go back to 1996.[57] Also similar are the eComma[58] and CATMA[59] commenting programs for humanists.

Another precursor of standoff properties is LMNL (layered markup annotation language).[60] This extends the basic model by adding annotations on ranges. This simple measure enables the nearly lossless importation of existing XML files, since each attribute consists of a name-value pair, which turns into the name-value of the LMNL annotation. These "annotations," or former attributes, can sometimes be merged with the main property name. For example, in TEI-XML the element "<hi rend="italic">…</hi>" can simply become the property

---

[54] G. Tummarello, C. Morbidoni, E. Pierazzo, "Toward textual encoding based on RDF" (paper presented at the ELPUB2005 Conference on Electronic Publishing – Kath. Univ. Leuven – June 2005); A. Di Iorio, S. Peroni, and F. Vitali, "A Semantic Web Approach to Everyday Overlapping Markup," *Journal of the American Society for Information Science and Technology,* 62.9 (2011): 1696-716.

[55] "SWRL: A Semantic Web Rule Language," <http://www.w3.org/Submission/SWRL/>.

[56] Di Iorio et al., *op. cit.,* p. 1706.

[57] M. Thaller, "Text as a Data Type," (paper presented at the ACH Conference, June 25-29, Bergen, 1996) <http://gandalf.aksis.uib.no/allc-ach96/Panels/Thaller/thaller2.html>.

[58] T. Brown, "eComma," <http://ecomma.cwrl.utexas.edu/e392k/>.

[59] "CATMA – Computer Aided Textual Markup and Analysis," <http://www.catma.de/>.

[60] W. Piez, "Towards Hermeneutic Markup: An architectural outline" (paper presented at the Digital Humanities Conference July 7-10, 2010).

“italics”. More complex attributes, however, such as links to other texts, will have to remain. It is the responsibility of the person who added those attributes to make use of them in standoff properties form, just as it was their responsibility to handle them in XML.

Standoff properties and/or the underlying text can be easily edited by the use of relative rather than absolute offsets.[61] With relative offsets the position of the property in the underlying text is described as its relative distance to the immediately preceding property in the set. In this way any changes to the set during editing will be localised, so only a small number of properties will be affected by any change to the underlying text.

The biggest advantage of standoff properties is its divide-and-conquer approach to markup. Since sets can be freely merged, markup can be separated into layers and recombined to reflect changing uses. This increased flexibility is in sharp contrast to embedded XML, where all the tags must go into one encoding and conform to a single hierarchy at a time.

4.3. Formatting Standoff Properties

Standoff properties can overlap freely, especially if two sets, stripped from separate well-formed XML documents, are merged. To render such combinations in a browser, however, requires them to be converted into a strict hierarchical structure, which they do not have. Although properties may overlap they mostly have an approximate nesting structure. In addition, the nesting of properties that will be converted into HTML is already known: it is simply the syntax of HTML. Combining these two sources of information furnishes sufficient hierarchical structure to allow a simple heuristic: if two properties overlap and one is higher up the hierarchy than the other, then split the one lower down, and make it the child of the higher property. In this way a HTML document tree can be quickly and correctly built, one property at a time.

The formatter tool[62] already performs this task by using custom CSS (cascading style sheet) rules, for example:

```
span.italics { font-style: italic }
```

This simple CSS rule states that all occurrences of the property “italics” are to be transformed into the HTML element “”. Any formats defined for that class are simply applied as usual by the browser, so the text will actually be in italics. An example using an annotation would be:

```
span.merged { -hrit-mergeid: id }
```

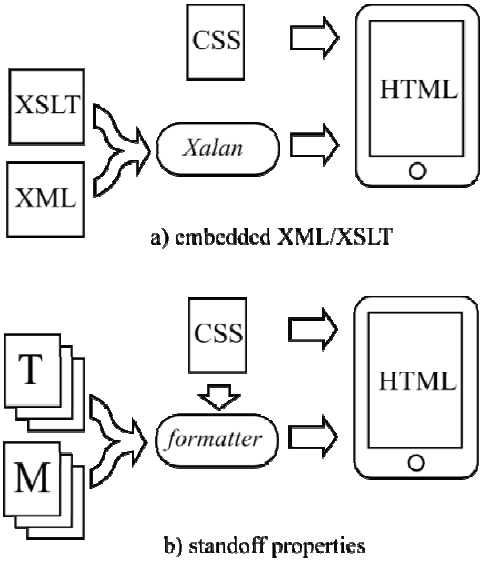
[61] T. H. Nelson, “Embedded Markup Considered Harmful,” <http://www.xml.com/pub/a/w3j/s3.nelson.html>.

[62] “hrit,” <http://code.google.com/p/hrit/source/checkout?repo=standoff>.

This uses a custom CSS property (starting with "-hrit-") to state that markup properties called "merged" with the annotation "mergeid" will be converted into HTML "" elements with an "id" attribute having the same value as the annotation.

In this way standoff properties can be converted into HTML without using XSLT,[63] so avoiding XML altogether. Since almost every HTML file already has an associated CSS file, no extra stylesheet is needed. The formatter tool thus converts the standard process that transforms an XML text using an XSLT stylesheet via a tool like Xalan[64] (Figure 2a), into one that combines a multi-version plain text and multi-version markup into a variety of possible HTML files, as shown in Figure 2b.

Figure 2: Converting to HTML



4.4. Loss of Standard Tools

One of the common objections to proposals for a non-XML system for digital humanities texts is the loss of standard XML tools. But what exactly are these tools, and how indispensible is their functionality? Only five tools appear to come into question:

1) XML Parsers
2) XML Editors

[63] J. Clark, "XSL Transformations (XSLT) Version 1.0," <http://www.w3.org/TR/xslt>.
[64] "Apache Xalan," <http://xalan.apache.org/>.

3) XSLT
4) XPath-based searching
5) XQuery

1-2) The first two tools check the syntax of an XML text. But there is no syntax in the original analog document that the digital surrogate represents. Syntax-checking is thus a function that serves XML, not the user. The user has no need of syntax, except to enable his/her files to be processed by XML tools. Likewise the display of attributes and context sensitive editing only serves the needs of XML. So the argument in favour of these kinds of tools is entirely circular.

3) The third tool for the most part is used to transform XML digital surrogates into HTML. Much of the XSLT language functionality is directed towards navigating XML, its complex elements and attributes, or servicing the syntax of XSLT. Only a small part actually generates the HTML. Although this is an important function, it could be performed in another way more simply and is not indispensible.

4) XPath search expressions are very useful if you have XML documents to search. Since searching tools can operate quite well on plain text, or on text structured in other ways, e.g. standoff properties, there seems no loss here at all, and perhaps even a gain in simplicity.

5) XQuery is used for extracting portions of XML documents using XPath expressions.[65] The formatter program can also extract parts of documents using properties. Although not yet as powerful as XQuery this facility could be expanded in response to user requirements.

So it seems the objection that a non-XML system would lead to a loss of standard tools is more about the discomfort of changing toolsets, than about the loss of anything that is irreplaceable.

4.5. A Digital Commons for Humanists

What software can we share? For surely every project differs in its goals and in the tools it uses. Although the development of graphical web interfaces is an idiosyncratic process, and will differ for each and every digital edition or knowledge site, there are still some common functions that many digital archives could share. Just as the Apache commons[66] provides an array of free software for building applications, what humanists need is a digital humanities commons. But first digital surrogates must be freed from their embedded dependencies on the end-result and subjectivity. This commons could provide:

[65] W3C, "XQuery 1.0: An XML Query Language (Second Edition),"<http://www.w3.org/TR/xquery/>.
[66] "Apache Commons," (Accessed April 19, 2012), <http://commons.apache.org/>.

1) Comparison of versions
2) Search and indexing
3) Web-editing tools
4) Conversion of data formats (import/export of foreign formats)
5) OCR tools especially for foreign language texts
6) Formatting for online display

No doubt more could be added, but these are functions that can be implemented and shared without reference to specific development environments, for example, content management systems, which are often used to build websites.

4.6. Importation of XML Data

A radical new model of markup for digital humanists would be of no use if existing documents in XML could not be imported. Standoff properties support LMNL style annotations, so any XML document can be converted into separate text and standoff properties files.

However, some caveats apply. Firstly, the problems associated with stripping tags from an XML file, as described above, will apply. Secondly, if alternatives have been defined, the XML file will first need to be split into several layers, which can later be merged into a single multi-version document. For example, if a text contained <app> and <rdg> elements, each unique version would need its own XML file with all <app> and <rdg> tags removed. Each version could then be stripped as normal.

Markup is also separated into versions, so that differences in formatting can later be found, even where the text is the same.

Exporting back to XML is possible because the order of tags is preserved on import. However, if any changes have been made, then the exported XML will only be an approximation of the more powerful standoff properties representation. Export, however, is not yet available in the HritServer application,[67] which is the main implementation of the design described here.

5. Conclusions and Future Work

Markup has always evolved both outside and within the digital humanities. The current form of embedded XML markup in use by digital humanists has problems relating to reusability, interoperability and tractability, among others. The recent planned development of large-scale digital repositories has brought these pre-existing problems out into the open. Unless both markup and text can be given a more flexible and interoperable form these problems threaten the viability of the new digital repositories.

[67] "hrit," <http://code.google.com/p/hrit/source/browse?repo=hritserver>.

The six problems outlined above are all solved by the proposed redesign:

1) Size is dealt with by subdividing the markup into separate sets, each reflecting one perspective of the markup. Such sets can then be merged as required for particular applications.
2) Usability is increased by separating markup from the text. The human editor need not see any markup codes when editing plain text; and since markup has been reduced to simple ranges, these could be edited by simply selecting and naming portions of text, as in CATMA.[68]
3) Overlap is solved by using independent tags without any overall syntax.
4) Interlinking is not necessary since tags may freely overlap. This also solves the tractability sub-problem (section 3.4.1 above).
5) Variation is dealt with by automatically merging texts rather than using manual markup.
6) Interoperability is increased by separating markup from the text. Plain text is highly interoperable; although markup tags are themselves not interoperable, separating them from the text greatly increases their flexibility because different sets may be freely combined.

5.1. Collaborations

Several research groups have already declared their support for this revision in the design of markup for the digital humanities.

The Digital Variants archive at Roma Tre, Italy[69] has a collection of genetic texts of modern works surviving in the form of written drafts. It was the representation of this material that provided the original impetus to develop the multi-version document concept in 2005.[70]

The HRIT (humanities, resources, infrastructure and tools) project at Loyola University, Chicago, is focused on print editions such as Thomas Hardy.[71] HRIT has given the project a more comprehensive scope[72] and also contributed to the development of the standoff properties markup model.

The Australian Electronic Scholarly Editing Project at the University of Queensland aims to develop a set of interoperable services, in collaboration with HRIT, to support the production of electronic scholarly editions by distributed collaborators in a Web 2.0 environment.[73] The textual surrogates within

---

[68] "CATMA – Computer Aided Textual Markup and Analysis," <http://www.catma.de/>.

[69] "Digital Variants," <http://www.digitalvariants.org>.

[70] D. Schmidt and T. Wyeld, "A novel user interface for online literary documents," *ACM International Conference Proceeding Series* 122 (2005): 1-4.

[71] "Center for Textual Studies in the Digital Humanities," <http://www.ctsdh.luc.edu/node/24>.

[72] S. E. Jones, P. Shillingsburg, G. Thiruvathukal, "Creative Engagement with Creative Works: a New Paradigm for Collaboration" (paper presented at the Digital Humanities Conference, King's College London, London July 7-10, 2010).

[73] "AustESE," <http://itee.uq.edu.au/~eresearch/projects/austese/>.

that environment will be stored using standoff properties and multi-version documents. The annotation system will be based on the Lore tool developed by AustLit, which uses the Open Annotation Collaboration model.[74]

References

Barnard, D., R. Hayter, M. Karababa, G. Logan, and J. McFadden. 1988. SGML-Based Markup for Literary Texts: Two Problems and Some Solutions. *Computers and Humanities* 22: 265-76.

Bauman, S. 2011. Interoperability vs. Interchange. *Balisage Series on Markup Technologies* 7, <doi:10.4242/BalisageVol7.Bauman01>, (Accessed April 16, 2012).

Bauman, S. 2005. TEI Horsing Around. Paper presented at the Extreme Markup Languages conference 1-5 August, Montreal, Canada.

Bourdaillet, J., and J.-G. Ganascia. 2006. MEDITE: A Unilingual Textual Aligner. Paper presented at the 5th International Conference on Natural Language Processing, 23-25 August, Turku, Finland. *LNAI* 4139: 458-69.

Clark, J. 1999. *XSL Transformations (XSLT) Version 1.0.* <http://www.w3.org/TR/xslt>.

DeRose, S. 2004. Markup Overlap: A Review and a Horse. Paper presented at the Extreme Markup Languages Conference, 2-6 August, Montreal, Canada.

Di Iorio, A., S. Peroni, and F. Vitali. 2011. A Semantic Web Approach to Everyday Overlapping Markup. *Journal of the American Society for Information Science and Technology,* 62 (9): 1696-716.

Eggert, P. 2005. Text-encoding, Theories of the Text, and the 'Work-Site.' *Literary and Linguistic Computing* 20 (4): 425-35.

Fielding, R. T. 2000. Architectural Styles and the Design of Network-based Software Architectures. PhD diss., University of California, Irvine.

Francis, W. N, and H. Kucera. 1979. Brown Corpus Manual. Providence, Rhode Island, 1964. <http://khnt.hit.uib.no/icame/manuals/brown/index.htm>.

Gabler, H. W. 2007. The Primacy of the Document in Editing. *Ecdotica* 4: 207.

Garey, M. R., and D. S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* New York: W.H. Freeman.

Hockey, S. M., and I. Marriott. 1980. *Oxford Concordance Program Version 1.0 User's Manual*. Oxford: Oxford University Computing Service.

Hunter, J., T. Cole, R. Sanderson, and H. van de Sompel. 2010. The Open Annotation Collaboration: A Data Model to Support Sharing and Interoperability of Scholarly Annotations. Paper presented at the Digital Humanities Conference 2010, King's College London, London July 7-10.

[74] J. Hunter, T. Cole, R. Sanderson, H. Van de Sompel, "The Open Annotation Collaboration: A Data Model to Support Sharing and Interoperability of Scholarly Annotations," (paper presented at the Digital Humanities conference, King's College London, London July 7-10, 2010).

Ide, N., and C. M. Sperberg-McQueen. 1988. Proposal for Funding for An Initiative to Formulate Guidelines for the Encoding and Interchange of Machine-Readable Texts. <http://projects.oucs.ox.ac.uk/teiweb/Vault/SC/scg02.html>.

Jones, S. E., P. Shillingsburg, and G. Thiruvathukal. 2010. Creative Engagement with Creative Works: a New Paradigm for Collaboration. Paper presented at the Digital Humanities Conference, King's College London, London July 7-10.

Piez, W. 2010. Towards Hermeneutic Markup: An architectural outline. Paper presented at the Digital Humanities Conference, King's College, London, July 7-10.

Schmidt, D., and T. Wyeld. 2005. A novel user interface for online literary documents. *ACM International Conference Proceeding Series* 122: 1-4.

Schmidt, D. 2009. Merging Multi-Version Texts: a Generic Solution to the Overlap Problem. Paper presented at Balisage: The Markup Conference 2009. *Balisage Series on Markup Technologies,* 3. <doi:10.4242/BalisageVol3.Schmidt01>.

Schmidt, D. 2010. The Inadequacy of Embedded Markup for Cultural Heritage Texts. *Literary and Linguistic Computing* 25 (3): 337-56.

SGML. ISO 8879. Information processing – Text and office systems – Standard Generalized Markup Language (SGML). ISO, Geneva, 1986.

Silva, G., and C. Bellamy. 1968. *Some Procedures and Programs for Processing Language Data.* Clayton: Monash University.

Sperberg-McQueen, C. M. 1991. Text in the Electronic Age Textual Study and Text Encoding, with Examples from Medieval Texts. *Literary and Linguistic Computing* 6 (1): 34-46.

TEI. 1990. *Text Encoding Initiative, Guidelines for Electronic Text Encoding and Interchange*, ed. C. M. Sperberg-McQueen and L. Burnard. Chicago, Oxford.

Thaller, M. 1996. Text as a Data Type. Paper present at the ACH Conference, June 25-29, Bergen. <http://gandalf.aksis.uib.no/allc-ach96/Panels/Thaller/thaller2.html>.

Tognazini, B. 1992. *TOG on Interface.* Reading: Addison-Wesley.

Tummarello, G., C. Morbidoni, and E. Pierazzo. 2005. Toward textual encoding based on RDF. Paper presented at the ELPUB2005 Conference on Electronic Publishing, Kath. Univ. Leuven, June.

Ukkonen, E. 1995. Online Construction of Suffix Trees. *Algorithmica* 14: 249-60.